SIAM J. NUMER. ANAL.
Vol. 20, No. 4, August 1983

0036-1429/83/2004-0014 $01.25/0

ON SHAPE PRESERVING QUADRATIC SPLINE INTERPOLATION*

LARRY L. SCHUMAKER†

Abstract. In this paper we discuss the design of algorithms for interpolating discrete data using C^1-quadratic splines in such a way that the monotonicity and/or convexity of the data is preserved. The analysis culminates in an interactive algorithm which takes full advantage of the flexibility which quadratic splines permit.

1. Introduction. This paper is concerned with the numerical solution of the following interpolation problem:

Problem 1.1. Given points $t_1 < \cdots < t_n$ and values $\{z_i\}_1^n$, find s such that

$$s(t_i) = z_i, \qquad i = 1, 2, \cdots, n. \tag{1.1}$$

While there are many methods available for the solution of this problem, in this paper we are concerned only with methods which preserve the shape of the data. By this we mean that in those intervals where the data is monotone increasing or decreasing, s should have the same property. Similarly, in those intervals where the data is convex or concave, the same should be true of s.

Recently, several shape preserving methods for the solution of Problem 1.1 have appeared (see [5]–[8], [10]–[17], and references therein). One of these methods (cf. [12]–[13]) constructs s as a C^1 quadratic spline with knots at the data points $t_1, \cdots, t_n$, and with one additional knot in each subinterval (t_i, t_{i+1}), $i = 1, \cdots, n-1$.

The purpose of this paper is to give a general treatment of the use of quadratic splines for solving Problem 1.1. In particular, we shall show exactly when it is necessary to add knots to a subinterval, and where they can be placed. We then use this information to describe a numerical algorithm which can be used to take full advantage of the flexibility which quadratic splines permit. In contrast with the previously published algorithms which rely on an ad hoc scheme for selecting knots, our algorithm allows the user to adjust their locations.

The paper is divided into six sections. In § 2 we examine a simple Hermite interpolation problem involving quadratic polynomials, which provides the basis for the whole paper. The general and specific algorithms are presented in §§ 3 and 4, respectively. We conclude with numerical examples and remarks in §§ 5 and 6.

2. A Hermite interpolation problem. In this section we discuss a simple two-point Hermite interpolation problem which will be useful in solving Problem 1.1.

Problem 2.1. Let $t_1 < t_2$, and suppose z_1, z_2, s_1, s_2 are given real numbers. Find a function $s \in C^1[t_1, t_2]$ such that

$$s(t_i) = z_i, \qquad s'(t_i) = s_i, \qquad i = 1, 2. \tag{2.1}$$

The following lemma shows that in certain cases, Problem 2.1 can be solved by a quadratic polynomial.

LEMMA 2.2. *There is a quadratic polynomial solving Problem* 2.1 *if and only if*

$$\frac{s_1 + s_2}{2} = \frac{z_2 - z_1}{t_2 - t_1}. \tag{2.2}$$

* Received by the editors March 30, 1982.

† Center for Approximation Theory, Department of Mathematics, Texas A&M University, College Station, Texas 77843.

In particular, if (2.2) *holds, then*

$$s(t) = z_1 + s_1(t-t_1) + \frac{(s_2-s_1)(t-t_1)^2}{2(t_2-t_1)} \tag{2.3}$$

is a solution.

Proof. The proof is elementary. □

When condition (2.2) fails, it is not possible to solve Problem 2.1 using a quadratic polynomial. We now show, however, that it can always be solved using a quadratic spline with one (simple) knot. A *quadratic spline* is a function $s \in C^1(-\infty, \infty)$ such that for some points $x_1 < x_2 < \cdots < x_n$, s restricted to each subinterval defined by the x's reduces to a quadratic polynomial. The x's are called the *knots* of the spline (for a detailed treatment of splines, see e.g. Schumaker [18]).

LEMMA 2.3. *For every $t_1 < \xi < t_2$, there exists a unique quadratic spline s with a (simple) knot at ξ solving Problem* 2.1. *In particular, we can write*

$$s(t) = \begin{cases} A_1 + B_1(t-t_1) + C_1(t-t_1)^2, & t_1 < t < \xi, \\ A_2 + B_2(t-\xi) + C_2(t-\xi)^2, & \xi \leqq t < t_2, \end{cases} \tag{2.4}$$

with

$$\begin{gathered} A_1 = z_1, \quad B_1 = s_1, \quad C_1 = (\bar{s}-s_1)/2\alpha, \\ A_2 = A_1 + B_1\alpha + C_1\alpha^2, \quad B_2 = \bar{s}, \quad C_2 = (s_2-\bar{s})/2\beta, \end{gathered} \tag{2.5}$$

where

$$\begin{gathered} \bar{s} = s'(\xi) = \frac{2(z_2-z_1) - (\alpha s_1 + \beta s_2)}{(t_2-t_1)}, \\ \alpha = \xi - t_1, \qquad \beta = t_2 - \xi. \end{gathered} \tag{2.6}$$

Proof. It is easily checked that the function s defined in (2.4)–(2.6) is a quadratic spline (to check this, one must check that s and its first derivative are both continuous across the knot at ξ). It is also easy to verify that s satisfies the interpolation conditions (2.1). Finally, the uniqueness of s follows from well-known zero theorems on splines (see e.g. [18, Thm. 4.53]). □

We now discuss the shape of the interpolating functions given in Lemmas 2.2 and 2.3. First we treat the special case of Lemma 2.2.

LEMMA 2.4. *Suppose $s_1 \cdot s_2 \geqq 0$, and that* (2.2) *holds. Then the quadratic polynomial s in* (2.3) *which solves Problem* 2.1 *is monotone on $I = [t_1, t_2]$. Moreover, if $s_1 < s_2$, then s is convex on I, while if $s_1 > s_2$, then s is concave on I.*

Proof. Clearly s' is linear on I. It follows that $s'(t)$ has the same sign as s_1 and s_2 throughout I. This establishes the monotonicity assertion. The assertion about convexity (concavity) follows immediately from the fact that $s''(t) = (s_2-s_1)/(t_2-t_1)$. □

In the remainder of this section we suppose that (2.2) fails, and discuss the shape of the spline s in (2.4). Clearly a necessary condition for monotonicity of s is that $s_1 \cdot s_2 \geqq 0$. We now give a sufficient condition.

LEMMA 2.5. *Suppose that $s_1 \cdot s_2 \geqq 0$. Then the spline* (2.4) *is monotone on $I = [t_1, t_2]$ if and only if $s_1 \cdot \bar{s} \geqq 0$. This condition can also be written as*

$$2(z_2-z_1) \geqq (\xi-t_1)s_1 + (t_2-\xi)s_2 \quad \text{if } s_1, s_2 \geqq 0, \tag{2.7}$$

$$2(z_2-z_1) \leqq (\xi-t_1)s_1 + (t_2-\xi)s_2 \quad \text{if } s_1, s_2 \leqq 0. \tag{2.8}$$

Proof. Since s' is piecewise linear, $s'(t)$ has one sign throughout I if and only if s_1, s_2, and $\bar{s}$ all have the same sign. □

Our next lemma deals with the convexity of s.

LEMMA 2.6. *Suppose that $s_1 < s_2$. Then the spline s in* (2.4) *is convex on $I = [t_1, t_2]$ if and only if*

$$s_1 \leqq \bar{s} \leqq s_2. \tag{2.9}$$

Similarly, if $s_1 > s_2$, then s is concave on I if and only if

$$s_2 \leqq \bar{s} \leqq s_1. \tag{2.10}$$

Proof. Since

$$s''(t) = \begin{cases} \dfrac{\bar{s} - s_1}{\xi - t_1}, & t_1 \leqq t < \xi, \\ \dfrac{s_2 - \bar{s}}{t_2 - \xi}, & \xi \leqq t \leqq t_2, \end{cases}$$

the assertion is obvious. □

Although Lemma 2.3 shows that the two-point Hermite interpolation problem can be solved by a quadratic spline with one knot placed arbitrarily in the interval (t_1, t_2), it is not possible to satisfy conditions (2.9) and (2.10) for arbitrary knot locations. The following lemma shows exactly which knot locations lead to convex or concave splines.

LEMMA 2.7. *Let $\delta = (z_2 - z_1)/(t_2 - t_1)$. Then $(s_2 - \delta)(s_1 - \delta) \geqq 0$ implies that s must have an inflection point in the interval I. Suppose now that $(s_2 - \delta)(s_1 - \delta) < 0$. Then the condition $|s_2 - \delta| < |s_1 - \delta|$ implies that for all ξ satisfying*

$$t_1 < \xi \leqq \bar{\xi} \quad \text{with} \quad \bar{\xi} = t_1 + \frac{2(t_2 - t_1)(s_2 - \delta)}{(s_2 - s_1)}, \tag{2.11}$$

the interpolating spline s in (2.4) *satisfies*

$$\begin{aligned} &s \text{ is convex on } I \text{ if } s_1 < s_2, \\ &s \text{ is concave on } I \text{ if } s_1 > s_2. \end{aligned} \tag{2.12}$$

If $s_1 s_2 \geqq 0$, then s is also monotone. Similarly, if $|s_2 - \delta| > |s_1 - \delta|$, then for all ξ satisfying

$$\underline{\xi} \leqq \xi < t_2 \quad \text{with} \quad \underline{\xi} = t_2 + \frac{2(t_2 - t_1)(s_1 - \delta)}{(s_2 - s_1)}, \tag{2.13}$$

the interpolating spline s satisfies (2.12). *If $s_1 s_2 \geqq 0$, then s is also monotone.*

Proof. It is easy to check that if $(s_2 - \delta)(s_1 - \delta) \geqq 0$, then conditions (2.9)–(2.10) fail, and s cannot be convex or concave on I. Now for the converse, consider the case $s_1 < s_2$ and $|s_2 - \delta| < |s_1 - \delta|$. First we note that in this case $t_1 < \bar{\xi} < t_2$. Now some computation shows that if $t_1 < \xi \leqq \bar{\xi}$, then (2.9) follows, which by Lemma 2.6 assures that s is convex on I. This, coupled with Lemma 2.5, shows that s is monotone when $s_1 s_2 \geqq 0$. The other cases are similar. □

3. A general curve fitting algorithm. We now describe a general curve fitting algorithm for solving the interpolation Problem 1.1 using quadratic splines.

ALGORITHM 3.1.

1. (INPUT)
   n = number of data points
   $\{t_1\}_1^n$, $\{z_i\}_1^n$, data
2. Either input or compute $\{s_i\}_1^n$
3. For $i = 1$ step 1 until $n-1$
   Use Lemma 2.2 to decide if a knot is needed in the interval $I_i = [t_i, t_{i+1}]$, and insert one if needed.
   Use Lemma 2.3 to compute the coefficients of the polynomial pieces associated with each knot.
4. (OUTPUT)
   k = number of knots
   $\{x_i\}_1^k$, the knots
   $\{A_i, B_i, C_i\}_1^k$, the coefficients of the polynomial pieces.

Discussion. This algorithm produces the coefficients of a piecewise-polynomial representation of a quadratic spline s solving Problem 2.1. In particular, we have

$$s(t) = \{A_i + B_i(t-x_i) + C_i(t-x_i)^2, \qquad x_i \leqq t < x_{i+1}, \qquad i = 1, 2, \cdots, k. \tag{3.1}$$

This is a convenient form for storing s. The values of s and its derivatives at any point t are easily calculated using Horner's scheme. □

We now discuss how this algorithm can be made to produce a quadratic interpolating spline which preserves the shape of the data. Throughout the following discussion we shall use the notation $I_i = [t_i, t_{i+1}]$ and

$$\delta_i = \frac{z_{i+1} - z_i}{t_{i+1} - t_i}, \qquad i = 1, 2, \cdots, n-1.$$

If a knot has been inserted in the interval I_i, we denote it by ξ_i.

Monotonicity. Lemmas 2.4 and 2.5 show how Algorithm 3.1 can be made to produce a spline s which is *locally monotone*. In particular, to guarantee that s is monotone on I_i, we must first make sure that $s_i \cdot s_{i+1} \geqq 0$. In addition, if a knot ξ_i is inserted in the interval I_i, then we must also require that

$$\bar{s}_i = \frac{2(z_{i+1} - z_i) - (\xi_i - t_i)s_i - (t_{i+1} - \xi_i)s_{i+1}}{(t_{i+1} - t_i)} \tag{3.2}$$

has the same sign as s_i and s_{i+1}. If $(s_i - \delta_i)(s_{i+1} - \delta_i) \geqq 0$, in order to insure (3.2) we must restrict the size of s_i and s_{i+1}, depending on the location of ξ_i. In particular, we need

$$2|z_{i+1} - z_i| \geqq |(\xi_i - t_i)s_i + (t_{i+1} - \xi_i)s_{i+1}|. \tag{3.3}$$

These conditions show how to make s monotone in the interval I_i. If the data is *globally monotone*, i.e., $z_1 < z_2 < \cdots < z_n$, then by selecting the slopes $\{s_i\}_1^n$ correctly, we can make s globally monotone also.

Co-monotonicity. Often a data set will not be globally monotone, but instead switches back and forth between monotone increasing and monotone decreasing. Because of the local nature of the quadratic spline s constructed in Algorithm 3.1, by choosing the slopes correctly, we can make s follow the shape of the data in the following sense:

$$\begin{aligned} &s \text{ is monotone increasing on } I_i \text{ if } z_i < z_{i+1}, \\ &s \text{ is monotone decreasing on } I_i \text{ if } z_i > z_{i+1}. \end{aligned} \tag{3.4}$$

When s has this property, we say that s is *co-monotone with the data.* It is clear that in order to achieve co-monotonicity, we must choose the slopes to assure local monotonicity, and in addition, we must insure that

$$s_i = 0 \quad \text{when } \delta_{i-1} \cdot \delta_i \leqq 0. \tag{3.5}$$

Convexity. Lemmas 2.6 and 2.7 can be used to make Algorithm 3.1 produce a spline s which is locally convex or concave in intervals I_i with $(s_i - \delta)(s_{i+1} - \delta_i) < 0$. In particular, to make s convex on the interval I_i, we need only make sure that condition (2.9) holds, while for concavity we need condition (2.10). These conditions can be guaranteed by choosing the knot ξ_i in the interval I_i, according to Lemma 2.7, i.e., satisfying

$$t_i < \xi_i \leqq t_i + \frac{2(t_{i+1} - t_i)(s_{i+1} - \delta_i)}{(s_{i+1} - s_i)} \quad \text{if } |s_{i+1} - \delta_i| < |s_i - \delta_i|, \tag{3.6}$$

$$t_{i+1} + \frac{2(t_{i+1} - t_i)(s_i - \delta_i)}{(s_{i+1} - s_i)} \leqq \xi_i < t_{i+1} \quad \text{if } |s_{i+1} - \delta_i| > |s_i - \delta_i|, \tag{3.7}$$

respectively. If the data is *globally convex*, i.e., $\delta_1 < \delta_2 < \cdots < \delta_n$, then by choosing $s_1 < s_2 < \cdots < s_n$ in such a way that s is locally convex in each subinterval, it will follow that s is globally convex. (A similar assertion holds for global concavity.)

Co-convexity. A data set may switch back and forth between convexity and concavity. In this case, it would be desirable to make s have a similar behavior. We can accomplish this by making s be locally convex or concave as needed. For example, suppose that $\delta_1 < \cdots < \delta_l > \delta_{l+1} \cdots > \delta_{n-1}$. Then by choosing the slopes such that $s_1 < s_2 < \cdots < s_l$, $s_{l+1} > \cdots > s_{n-1}$ and the conditions (3.6) are satisfied for $i = 1, 2, \cdots, l-1$, while (3.7) are satisfied for $i = l+1, \cdots, n-1$, we obtain a spline such that s is convex on $[t_1, t_l]$ and concave on $[t_{l+1}, t_n]$. In this case we say that s is *co-convex* with the data. (Note that in this example, s must have an inflection point in the interval $[t_l, t_{l+1}]$—indeed, otherwise it could not switch from convex to concave.)

Linear segments. Sometimes a number of consecutive data points will lie on a straight line; e.g., we might have $\delta_l = \cdots = \delta_{r-1} = \delta$. In this case it may be desirable to insure that s not only goes through the data points (t_i, z_i), $i = l, \cdots, r$, but also to insure that s is in fact linear on $[t_l, t_r]$. This can be accomplished by choosing the slopes such that $s_l = \cdots = s_r = \delta$.

We conclude this section by observing two further properties of Algorithm 3.1. Both are due to the local nature of the quadratic spline which is being constructed.

Adjusting one data value. If a spline fit s has been calculated for a given set of data $\{t_i, z_i\}_1^n$, and it is desired to change the ith data point (by changing z_i and/or by moving t_i to another point between t_{i-1} and t_{i+1}), then it is not necessary to recalculate all of the coefficients of a new spline fit. Indeed, we need only consider the interpolation problem on $[t_{i-1}, t_{i+1}]$, and then mesh the new knots and coefficients with the ones we already have.

Extending the data. There are some applications where the data comes to us in a continuous stream (for example, from telemetering equipment). In this case we cannot wait until all the data is in to perform a fit—instead we must perform a running fit. Algorithm 3.1 is well-suited for this task. Having computed a fit to the first n data, we can fit additional data without recomputing the knots and coefficients of the spline which has already been computed.

4. A specific algorithm. In this section we describe a specific algorithm for solving Problem 2.1 using quadratic splines. The algorithm is based on Algorithm 3.1, and

is designed to preserve the shape of the data. While it can be used as a one pass algorithm, it can also be integrated into an interactive package where it would serve as a first pass. The user then could interactively adjust the slopes and knot locations in order to alter the shape of the interpolating curve as desired. We say more about how to set up this interactive package later in this section.

ALGORITHM 4.1.

1. Preprocessing
   For $i = 1$ step 1 until $n-1$,
   $L_i = [(t_{i+1}-t_i)^2 + (z_{i+1}-z_i)^2]^{1/2}$
   $\delta_i = (z_{i+1}-z_i)/(t_{i+1}-t_i)$
   For $i = 1$ step 1 until $n-1$,
   $L_i = \sum_{l_i}^{r_i} L_j$, where $s_{l_{i-1}} \neq s_{l_i} = \cdots = s_{r_i} \neq s_{r_i+1}$
2. Slope calculations
   For $i = 2$ step 1 until $n-1$,
   $s_i = (L_{i-1}\delta_{i-1} + L_i\delta_i)/(L_{i-1}+L_i)$.
3. Left end slope
   $s_1 = (3\delta_1 - s_2)/2$
4. Right end slope
   $s_n = (3\delta_{n-1} - s_{n-1})/2$
5. Compute knots and coefficients
   $j = 0$.
   For $i = 1$ step 1 until $n-1$,
   if $s_i + s_{i+1} = 2\delta_i$
   $j = j+1$, $x_j = t_i$, $A_j = z_i$, $B_j = s_i$, $C_j = (s_{i+1}-s_i)/2(t_{i+1}-t_i)$
   else
   $a = s_i - \delta_i$, $b = s_{i+1} - \delta_i$
   if $ab \geqq 0$
   $\xi_i = (t_{i+1}+t_i)/2$
   else
   if $|a| > |b|$
   $\xi_i = t_{i+1} + a(t_{i+1}-t_i)/(s_{i+1}-s_i)$
   else
   $\xi_i = t_i + b(t_{i+1}-t_i)/(s_{i+1}-s_i)$
   $\bar{s}_i = (2\delta_i - s_{i+1}) + (s_{i+1}-s_i)(\xi_i - t_i)/(t_{i+1}-t_i)$
   $\eta_i = (\bar{s}_i - s_i)/(\xi_i - t_i)$
   $j = j+1$, $x_j = t_i$, $A_j = z_i$, $B_j = s_i$, $C_j = \eta_i/2$
   $j = j+1$, $x_j = \xi_i$, $A_j = z_i + s_i(\xi_i - t_i) + \eta_i(\xi_i - t_i)^2/2$
   $B_j = \bar{s}_i$, $C_j = (s_{i+1}-\bar{s}_i)/2(t_{i+1}-\xi_i)$.

Discussion. The slope s_i at the point t_i is computed as a weighted average of δ_{i-1} and δ_i, where the weights are taken to be the quantities L_{i-1} and L_i (which at this point have the lengths of the longest lines containing the chords in I_{i-1} and I_i, respectively). In particular, if $\delta_i = \delta_{i-1}$, then $s_i = \delta_i = \delta_{i-1}$. The slope s_1 is taken to be an average of δ_1 and s_2. Similarly, the slope s_n is taken to be an average of δ_{n-1} and s_{n-1}.

The choice of knots in those intervals where they are required is made in such a way as to assure local monotonicity and local convexity (or concavity). In particular, when $a = s_i - \delta_i$ and $b = s_{i+1} - \delta_i$ satisfy $ab \geqq 0$, then we are free to choose ξ_i anywhere in the interval I_i; we take the midpoint. When $ab < 0$, we take ξ_i to be the midpoint of the allowed interval (see Lemma 2.7 for the definition of this interval). Note that

in this case it may happen that condition (3.3) guaranteeing local monotonicity will fail. This can be corrected later in the interactive stage.

Algorithm 4.1 is designed to produce a quadratic spline with the following features:

- s is continuously differentiable.
- s is co-convex with the data, in the sense that it has inflection points only in those intervals where the data transitions from convex to concave or vice versa.
- s is co-monotone with the data with the possible exception of those intervals where s has an inflection point.

We conclude this section with some remarks on how Algorithm 4.1 can be integrated into an interactive system in which the user has considerable flexibility in adjusting the shape of the interpolating spline. The general structure of such an interactive system is shown in the flow chart in Fig. 1.

Adjusting slopes. The user is free to adjust the slope at each data point. In doing so, he should keep the following relationships between the value of s_i and the shape

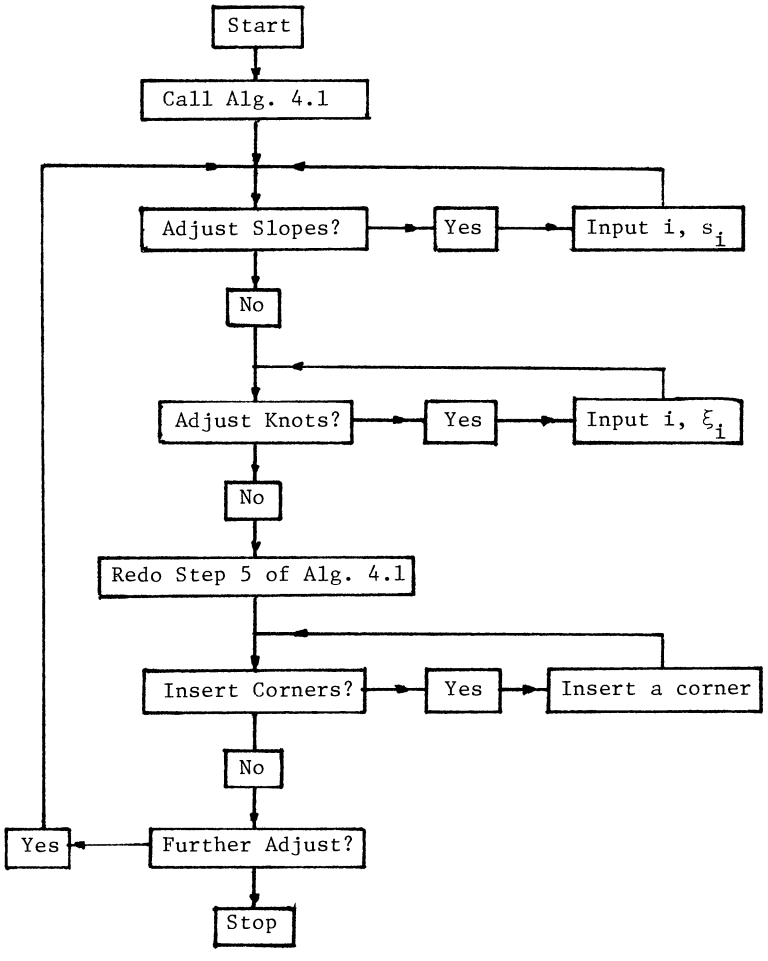

FIG. 1. *An interactive curve package.*

of s in mind:

- A necessary condition for monotonicity on I_{i-1} is that $s_i s_{i-1} \geqq 0$.
- A necessary condition for monotonicity on I_i is that $s_i s_{i+1} \geqq 0$.
- A necessary condition for monotonicity on both I_i and I_{i-1} in the case where $\delta_i \delta_{i-1} \leqq 0$ is that $s_i = 0$.
- Condition (3.3) is necessary to assure monotonicity on an interval I_i where s has an inflection point.

Adjusting knots. The user is also free to adjust the location of the knot ξ_i in each interval I_i where one is required. In choosing knots, he should keep the following facts in mind:

- ξ_i can be chosen anywhere in (t_i, t_{i+1}) when $a_i b_i \geqq 0$, where $a_i = s_i - \delta_i$ and $b_i = s_{i+1} - \delta_i$.
- ξ_i can be chosen anywhere in the interval (3.6) when $a_i b_i < 0$ and $|b_i| < |a_i|$.
- ξ_i can be chosen anywhere in the interval (3.7) when $a_i b_i < 0$ and $|b_i| > |a_i|$.

Inserting corners. Suppose that for some l, i, r, $\delta_l = \cdots = \delta_{i-1} \neq \delta_i = \cdots = \delta_r$. Then (t_i, z_i) is the point where the line drawn through the data points at $t_l, \cdots, t_i$ meets with the line through the data at $t_i, \cdots, t_r$. In some cases it may be desirable to make our fit s fit these lines exactly. Since Algorithm 4.1 will not produce such a fit (the slope at t_i is an average of δ_{i-1} and δ_i), if we want a corner at t_i, we must insert it interactively. This is a simple matter of dropping the knots in I_{i-1} and I_i, and adjusting a few coefficients. The quadratic spline which results is now only continuous at t_i, rather than differentiable. We can think of t_i as a double knot of the spline.

5. Numerical examples. Algorithm 4.1 and an interactive package based on the flow chart of Fig. 1 were prepared in FORTRAN and tested on a variety of numerical examples. In this section we present two typical data fitting problems to illustrate the flexibility of the package.

Example 5.1. Let $n = 5$, and suppose the t's and z's are given by

t	1	2	3	4	5
z	1	2	3	2	1

Discussion. The data and the spline s produced by Algorithm 4.1 are shown in Fig. 2a. Note that the slope at $t_3 = 3$ has been chosen to be 0, and that s is co-monotone with the data. In Fig. 2b we show the quadratic spline which results when the slope at t_3 is changed to 1, forcing the spline to be linear throughout $[1, 3]$. In this case, s is no longer monotone on $[3, 4]$. Finally, in Fig. 2c we show the quadratic spline s which results when we make the point t_3 be a corner (or double knot).

Example 5.2. Let $n = 11$, and suppose the t's and z's are given by

t	0	2	3	5	6	8	9	11	12	14	15
z	10	10	10	10	10	10	10.5	15	50	60	85

Discussion. This data is taken from Akima [1]. For the results of some other spline fits, see [10]. The data and the spline s produced by Algorithm 4.1 are shown

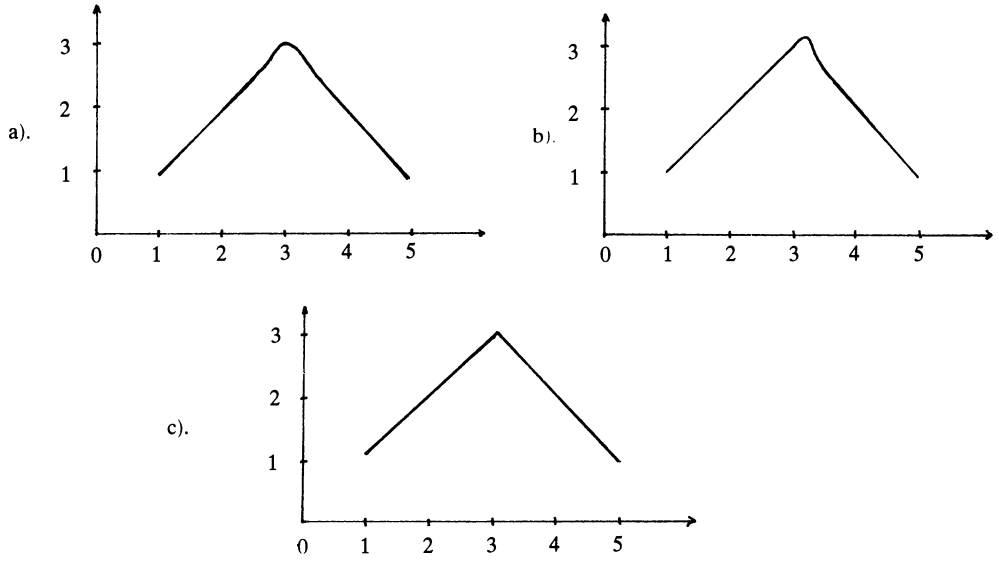

FIG. 2. *The splines in Example* 5.1.

in Fig. 3a. The slopes selected by the algorithm are

t	0	2	3	5	6	8	9	11	12	14	15
s	0	0	0	0	0	.061	1.92	30.96	28.23	19.21	27.85

The algorithm inserts knots at {7, 8.76, 10.977, 11.5, 13, 14.33}. Note that although the data is monotone increasing, the spline s fails to be monotone in the interval [12, 14]. This is due to the fact that the selected slopes at 12 and 14 are too large. Also note the slight dip in [7, 8].

In Fig. 3b we show the spline which results when the slopes at 12 and 14 are reduced to 11 and 8, respectively. This reduction is enough to assure that (3.3) is satisfied, and the spline becomes monotone increasing on [12, 14]. Its shape is changed only slightly outside this interval.

Fig. 3c shows the result of setting the slope at 8 to be 10. This forces the interpolating spline to be linear throughout the interval [0, 8].

6. Remarks.

1. Algorithm 4.1 presented here can be thought of as an alternative to the method of McAllister and Roulier [12]–[13]. The main differences between the two are in the way in which the slopes and knots are selected. Their slope and knot assignments are based on a geometrical argument, and the quadratic polynomial pieces are constructed using Bernstein polynomials. Whichever algorithm is used as a first pass, the main idea of this paper is that there is considerable flexibility in the use of quadratic splines, and that this flexibility should be exploited in an interactive mode.

2. In [12]–[13], there is some discussion of "pathological cases" where a decision based on comparing two numbers can make a substantial change in the shape of the fit. Algorithm 4.1 produces a spline fit s which depends continuously on the data. The location of a knot in an interval I_i may be a discontinuous function of the data.

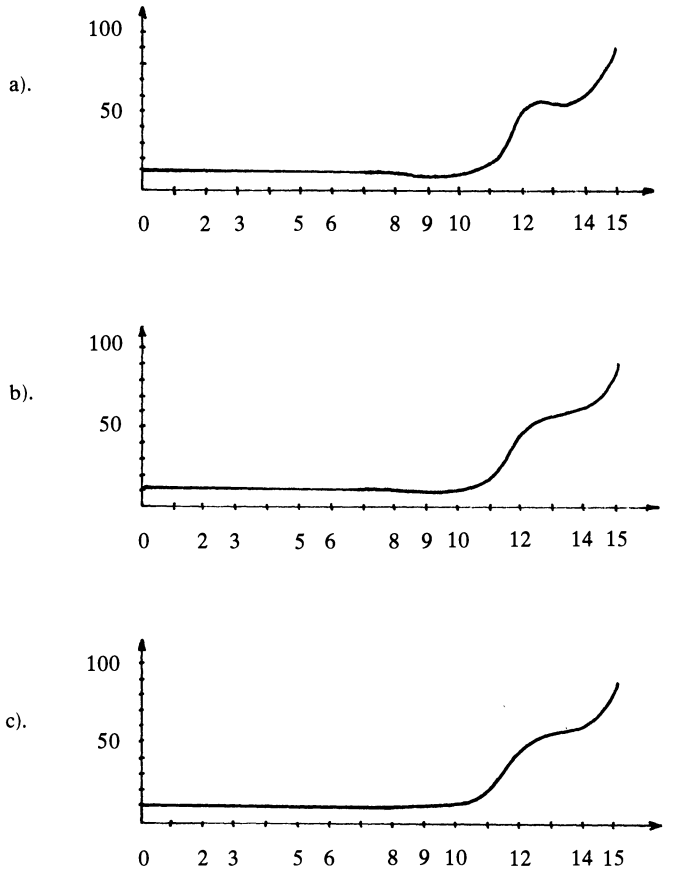

FIG. 3. *The splines in Example* 5.2.

In particular, if $(s_i - \delta_i)$ and $(s_{i+1} - \delta_i)$ are nearly the same, then the choice of a knot in I_i will depend on which of these is greater, and indeed, if they are equal, no knot will be inserted at all. Still, the final s will be nearly the same in all cases. I suggest that in coding the algorithm that the quantities be regarded as equal whenever they differ by no more than some prescribed small ε—this saves adding knots where they really are not needed.

3. Fritsch and Carlson [10] have published a curve fitting algorithm based on cubic splines, which is capable of producing monotone fits to monotone data. It does not preserve convexity, however.

4. Additional references and examples can be found in the papers listed below. For some unusual applications of shape preserving algorithms, see [6], [15].

REFERENCES

[1] H. AKIMA, *A new method of interpolation and smooth curve fitting based on local procedures*, J. Assoc. Comput. Mach., 17 (1970), pp. 589–602.

[2] C. DE BOOR, *A Practical Guide to Splines*, Springer-Verlag, New York, 1978.

[3] C. DE BOOR AND B. SWARTZ, *Piecewise monotone interpolation*, J. Approx. Theory, 21 (1977), pp. 411–416.

[4] J. BUTLAND, *A method of interpolating reasonable-shaped curves through any data*, Proc. Computer Graphics 80, Online Publications Ltd., Middlesex, UK, 1980, pp. 409–422.

[5] A. K. CLINE, *Scalar- and planar-valued curve fitting using splines under tension*, Comm. ACM, 17 (1974), pp. 218–223.

[6] L. E. DEIMEL, C. L. DOSS, R. J. FORNARO, D. F. MCALLISTER AND J. A. ROULIER, *Application of shape-preserving spline interpolation to interactive editing of photogrammetric data*, Proc. SIGGRAPH 78, 12 (1978), pp. 93–99.

[7] L. E. DEIMEL, D. F. MCALLISTER AND J. A. ROULIER, *Smooth curve fitting with shape preservation using osculatory quadratic splines*, Proc. 11th Annual Conference on the Interface Between Statistics and Computer Science, 1978, pp. 343–347.

[8] R. P. DUBE, *Univariate blending functions and alternatives*, Computer Graphics and Image Processing, 6 (1977), pp. 394–408.

[9] T. M. R. ELLIS AND D. H. MCLAIN, *Algorithm* 514. *A new method of cubic curve fitting using local data*, ACM Trans. Math. Software, 3 (1977), pp. 175–178.

[10] F. N. FRITSCH AND R. E. CARLSON, *Monotone piecewise cubic interpolation*, this Journal, 17 (1980), pp. 238–246.

[11] D. F. MCALLISTER AND J. A. ROULIER, *Interpolation by convex quadratic splines*, Math. Comput., 32 (1978), pp. 1154–1162.

[12]———, *An algorithm for computing a shape-preserving osculatory quadratic spline*, ACM Trans. Math. Software, 7 (1981), pp. 331–347.

[13] ———, *Algorithm* 574. *Shape-preserving osculatory quadratic splines*, ACM Trans. Math. Software, 7 (1981), pp. 384–386.

[14] D. F. MCALLISTER, E. PASSOW AND J. A. ROULIER, *Algorithms for computing shape preserving spline interpolations to data*, Math. Comp., 31 (1971), pp. 717–725.

[15] D. F. MCALLISTER, J. A. ROULIER AND M. EVANS, *Generation of random numbers using shape preserving quadratic splines*, Proc. 16th Ann. Southeast Regional ACM Conf. (1978), pp. 216–218.

[16] E. PASSOW, *Piecewise monotone spline interpolation*, J. Approx. Th., 12 (1974), pp. 240–241.

[17] E. PASSOW AND J. A. ROULIER, *Monotone and convex spline interpolation*, SIAM J. Numer. Anal., 14 (1977), pp. 904–909.

[18] L. L. SCHUMAKER, *Spline Functions: Basic Theory*, Wiley-Interscience, New York, 1981.

[19] S. W. YOUNG, *Piecewise monotone polynomial interpolation*, Bull. Amer. Math. Soc., 73 (1967), pp. 642–643.